NYU | STEINHARDT

# APSTA-GE 2047 Messy Data and Machine Learning

**Instruction: Ravi Shroff**

**Reported by:**

Adrian Harris

Snowy Chen

Spring 2022 Term, May 08, 2022

# Table of Contents

---

# I. ABSTRACT

RuneScape is a point-and-click game set in a fantasy world called Gielinor, where players can interact with each other (Lifewire, 2020). Every player decides their fate and can choose to do as they please, whether they want to train a skill, fight a monster, partake in a quest, play a mini-game or socialize with other players. What players do is entirely up to them, as everything is optional. Players learn skills in RuneScape through training. Different skills require different types of exercise, but they all follow the same basic formula: do something, gain experience, gain levels, gain abilities. This game has gone through three different stages: Runescape, Runescape 2, and Runescape 3. After a player vote, they decided to bring back Runescape 2 with is now renamed Old school Runescape. This was launched on February 22, 2013. This analysis will be focused on Runescape 3.

In this report, we want to explore which attributes of the RuneScape game contribute the most to the player's total XP score to find the best strategy for the players to place themselves in the higher rank on the leaderboard. There are many guides on how to play the game. This research is supposed to find a new ultimate strategy to play this game if a player wants to optimize their overall XP, which is a close proxy for their overall level as well.

# II. DATA

In this data section, we will introduce our data sources, how to pull the raw data, and the process of cleaning the raw dataset.

## A. Explore the Data

According to the research question, the values that we need are 1) player's total level, 2) total XP, 3) level for each skill, 4) XP for each skill, and 5) what skills they have. In RuneScape, page "HISCORES" allows us to access the ranking categories, where we can see all the required data but in separate links. The data from the "HISCORES" page are formatted in the table where we have columns of players' username, rank, total level, and total XP scores. When we click the row of a specific player, it will lead us to a different link: a table in this particular player, the table formatted with columns of rank, XP score, and level, and the row of each skill. Back to the "HISCORES" page, we can also click the "Skills" dropdown list, which will output each player's table with the skills we picked up from the dropdown list. There are 29 options in total, which are *"Attack", "Constitution", "Mining", "Strength", "Agility", "Smithing", "Defence", "Herblore", "Fishing", "Ranged", "Thieving", "Cooking", "Prayer", "Crafting", "Firemaking", "Magic", "Fletching", "Woodcutting", "Runecrafting", "Slayer", "Farming", "Construction", "Hunter", "Summoning", "Dungeoneering", "Divination", "Invention", "Archaeology", "Overall"*. The *"Overall"* option is the sum of all

individual skills. We noted from the comparison page that maximum level of each skill could be either 99 or 120.


## B. Process of Cleaning the Data

*Web scrapper - Attack.R.* Since the player's overall information table and individual skills table are in separate links, in order to combine the overall columns and individual skills columns and pull all player's usernames, we make an R file (Attack.R) that web scraped all the player's username from the "HISCORES" page, and create a new data frame that includes a column of player's username, rank, level, and XP. Then save the data frame as .csv formate with leaderboard as the suffix.

*Web scrapper - NewScrapper.R.* As we mentioned previously, the output data frame from Attack.R file only has the player's overall information but does not include each skill. In order to embed each skill to the data frame, we created another R file called NewScrapper.R, which will pull the one specific player's information from a different URL after we click this particular player from the "HISCORE" page. The other URL will lead us to the comparison page between two players with their skills, rank for each skill, and XP for each skill. The NewScrapper.R will web-scrapped all the data of each two players from the compare URL link. Since each compare URL only includes two players, we iterate 1k URLs (2k players) and then output the data frame with a column of XP score, level, rank, and player's usernames. The row of the output data frame will be each skill of the players. Since each iteration will only output 2k players, thus we will have to manually change the page number in order to output more player's data to have enough data points for later's model and analysis. Then we will save the data to .csv format with "NewData" as the prefix.

## 1. Feature engineering

*Clustering - EDA.R.* From the previous NewScrapper.R file, we pulled the data of each player from the comparison page and merged the two players from each compare URL into one data frame. We will apply the clustering algorithm to this dataset, which can interpret the input data and find the natural groups or clusters in the feature space. The [1]clustering algorithm will be applied to the raw dataset divided into four subgroups. As

[1] https://runescape.wiki/w/Combat_pure#Member_pures_(P2P)
_https://runescape.fandom.com/wiki/Skill_pure

we mentioned earlier, most of the skills go to the 99 maximum levels, but some go to 120.

In RuneScape, players can choose whichever strategy they prefer. Due to the different strategies that players choose, we found there are multiple types of players that can be identified due to their individual skill levels. We found multiple articles attached to the footnote below that define these types of players as pure. The combat pures includes *Basic Member's Pure* (60 Attack, 80+ Strength, 82+ Magic, 80+ Range, and 1 Defence); *Obby Mauler Pure* (1 Attack, 60+ Strength, 1 Defence); *Attack Pure* (60+ Attack, 1 Strength, 1 Defence); *Black Pure* (25 Defence, 60+ Attack, 80+ Strength, 75+ Constitution, 80+ Range, 82+ Magic); *Turmoil/Proselyte Pure* (60+ Attack, 99 Strength, 94 Magic, 80+ Range, 28 - 35 Defence, 95 Prayer); *Barrows Pure* (70+ Attack, 70+ Strength, 70 Defence, 94 Magic, 70+ Prayer); *Anti Pure* (99 attack, 1 Strength, 99 Defense, 1 or 52 or 99 Prayer, 1 or 50 of Magic. 1 or 50 of Range, 99 Constitution, 1 or 88 or 99 Summoning); *Summoning Tank/Defense Pure* (99 Defense, 1 Strength, 1 Attack, 1 Magic, 1 Range, 1 or 43 or 75 or 99 of Prayer, 99 Constitution, 1 or 99 Summoning); *Skill Pure* (all Combat Pure are level 1). In the EDA.R file, we also considered these pures as different variables and mutated these variables as individual columns. As we mentioned previously, the maximum level of each skill can be either 99 or 120. We also make two columns as *number_of_99s* which indicates the maximum level is 99, and *number_of_120s* which indicates the maximum level is 120. In the EDA.R file, we separated the players due to their skill level. Players with more than five skills levels are

equal to 99. We consider them as advanced players. On the contrary, for players with less equal than five skills' levels equivalent to 99, we consider them as elementary players. We separated these two groups of players into separate datasets *feature_eng_above_5_99s_data.csv*, which indicates the advanced players' dataset *feature_eng_below_5_99s_data.csv,* which indicates the dataset of the elementary players. When looking at the initial distributions of the data, we decided to separate the data on this condition to make sure the advanced-players aren't skewing the predictions.

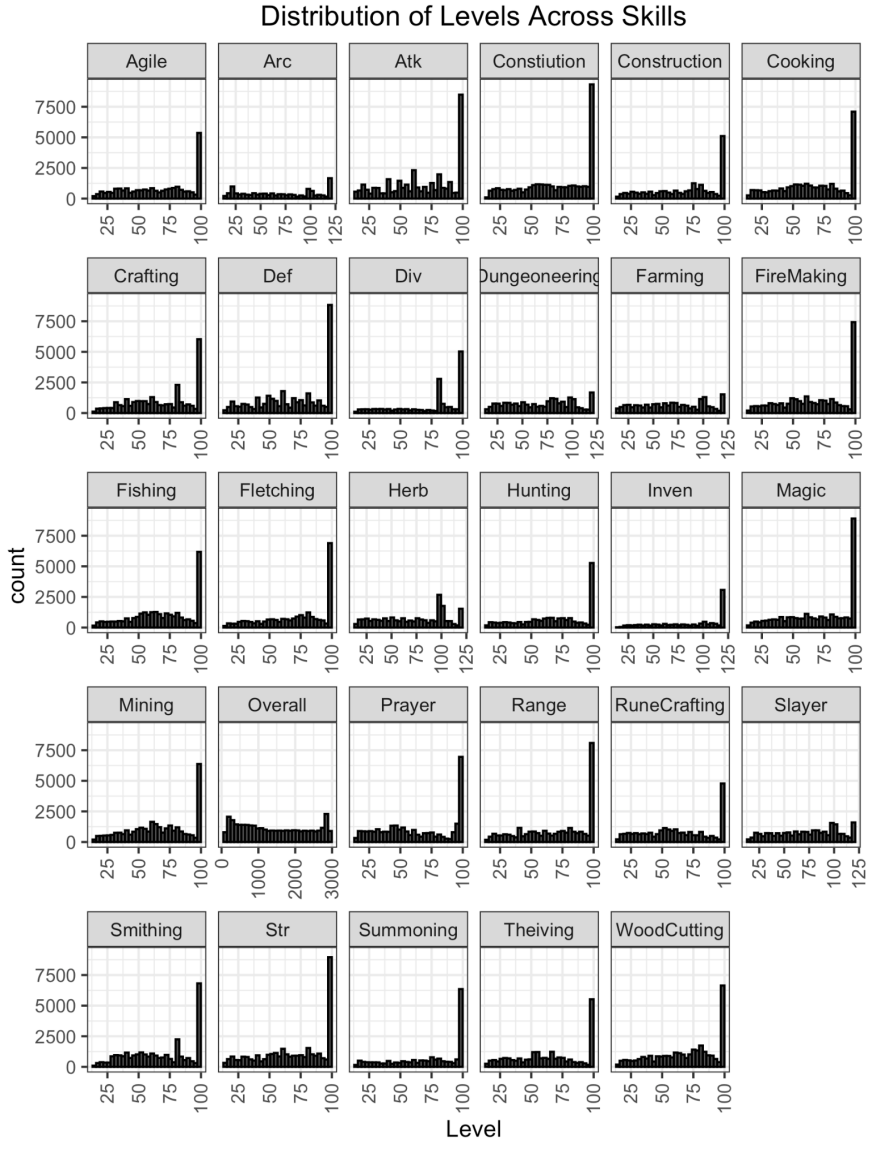

*Figure 1: Distribution of Level Across each Skills*

Worth noting that at the beginning of the EDA.R file, we take the output dataset from NewScrapper.R file, which are multiple datasets with column player's username, rank, XP score, and level. The row will be the skills of each player. In order to do the linear regression analysis in the following section, we also changed the format of these two datasets from long to wide. The output dataset structure is formatted in a table where we have a row for each player. As for the columns, we made each skill as separate columns like overall level, level of Attack, level of Defense, etc., plus the cluster and pures variables.

# III. ANALYSIS AND MODELS

## A. Linear Regression

We did a linear regression analysis on two output datasets from the EDA.R file, which are advanced players and elementary players.

*Linear Regression - Modeling_above.R and Modeling_below.R.* In the linear regression analysis section, we have two R files which are *Modeling_above* and *Modeling_below*. The First R file tends to do the linear regression analysis on the advanced player and the second R file is responsible for the elementary players. In order to predict the relationship between the total XP scores and each skill, firstly, we separated both datasets into training and test datasets and predicted the total_xp_overall as the function of all other variables

in the advanced players' dataset. According to the output of the Modeling_above.R file and the Modeling_below.R file, we can see that the Mean Squared Error (MSE) are 374268987320694528.00 and 720401752115511.50, which are extremely large. This also indicates that the linear regression analysis did a lousy job explaining our dataset.

## B. Xgboost

*Xgboost - Modeling_above_5_99s.ipynb & Modling_below_5_99s_ipynb.* The second machine learning model that we choose to use is Xgboost. Like linear regression, we use the advanced-player and elementary-player datasets to do the Xgboost analysis separately. For the reason that R crashed down every single time we tried to run the Xgboost script, we had to change to the Python script for the Xgboost model. We learned the SHAP value from this project that we had never got involved with it before. The SHAP value can be either positive or negative, which indicates that it's affect our model positively or negatively. To put it in another way, SHAP value has the similar purpose as the coefficient from the linear regression. The positive SHAP value works like the positive coefficient from linear regression, same for the negative SHAP value. In the following two figures, the x-axis stands for SHAP value, and the y-axis has all the features. Each point on the chart is one SHAP value for a prediction and feature. Red color means higher value of a feature. Blue means lower value of a feature. We can get the general sense of features' directionality impact based on the distribution of the red and blue dots.

# IV. Results

## A. Results Discussion

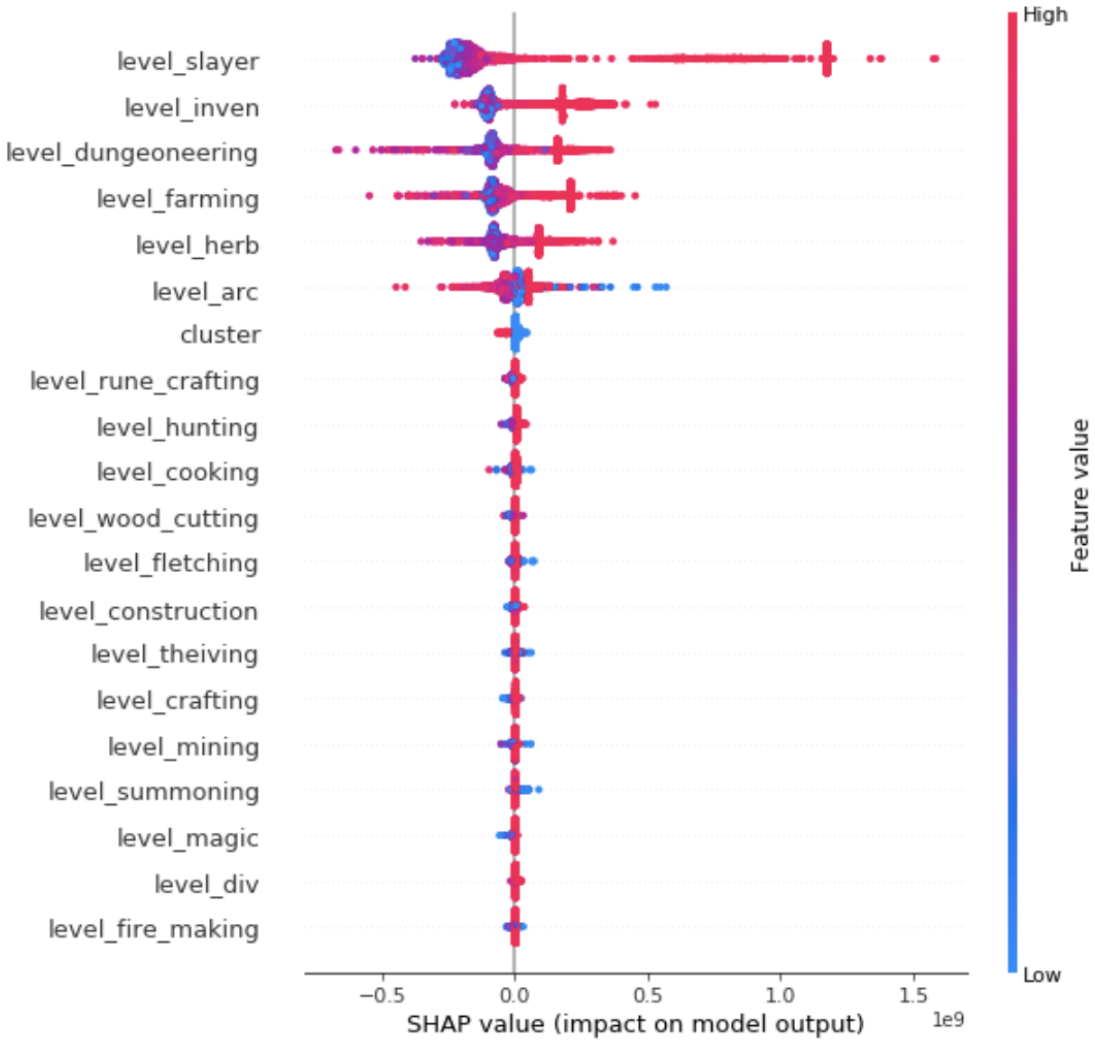


*Figure 2: SHAP value Advanced-player*

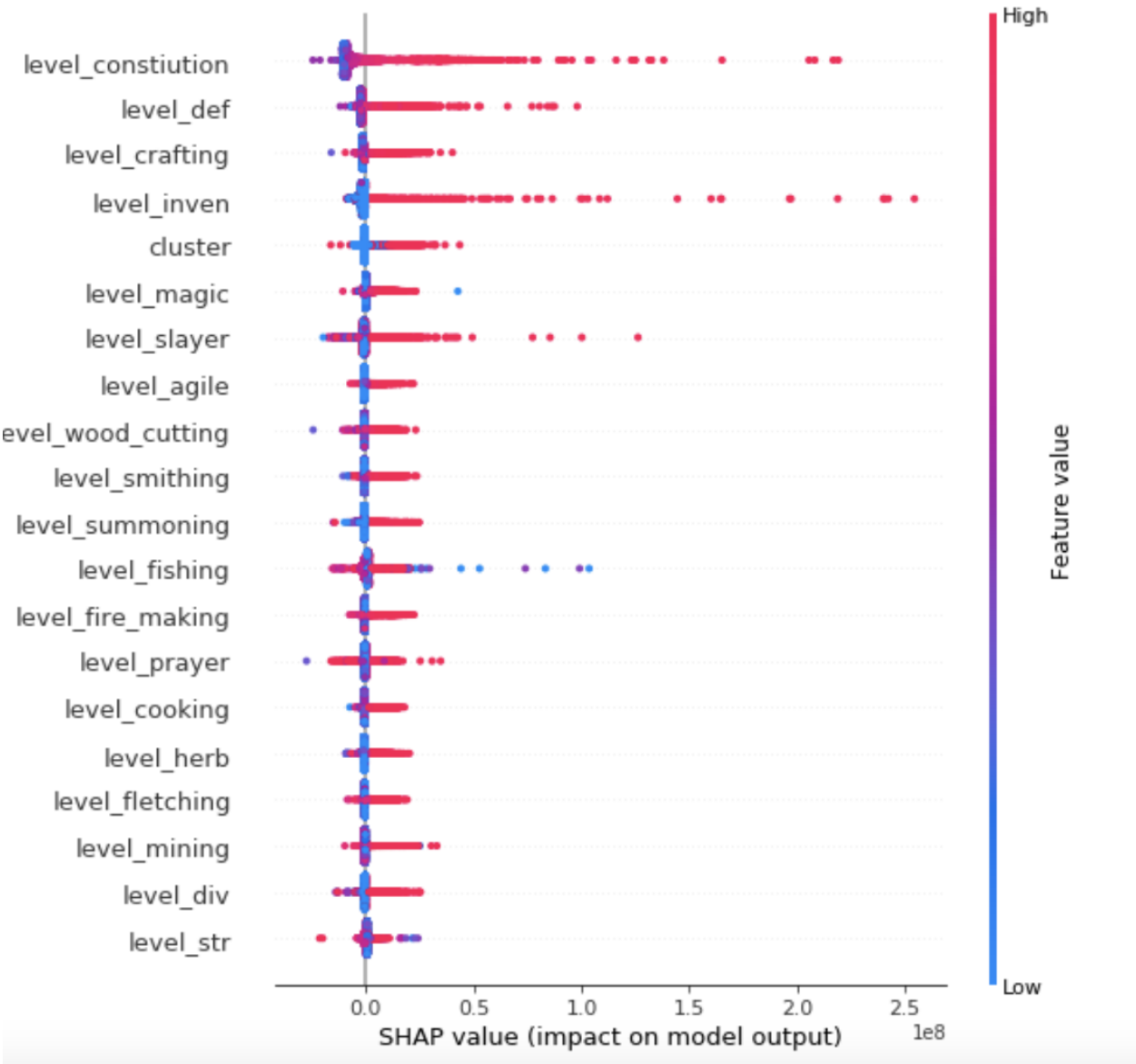


*Figure 3: SHAP value Elementary-Player*

When looking at what variables affect the SHAP values the most, we see variables such as *Slayer, Invention, Dungeonering , Farming*, and *Herblore* show up in the above 99 players. These skills take up time or resources, making sense that higher-level players would have spent the time and have the resources to level up most of these skills. In the elementary-player skills such as *Consitution, Defence, Crafting, Invention*, and the cluster

that was made were variables that influenced the predictions of overall XP the most. Skills such as *Consitution, Defence*, and *Crafting* are cheaper to level up and don't require as many resources. We believe that these results aren't the best due to not enough observations of what is possible to have (500k >). We also believe there isn't enough variance to separate in our outcome variable to really make accurate predictions. This problem can be seen with very high MSE from the linear regression.

## B. Limitations

Although the predicted results are under the expectation due to our group member's rich experiences in the RuneScape game, regardless, there are limitations in data clearing and the modeling analysis, and we will talk about these limitations in detail in the following sections.

### 1. Web scrapping

Unlike our previous experiences that work with the data that has already been provided, we need to manually make the script to web scrap the data of all players and other variables that we need.

The first lamination that we run into is the IP address issue. In the "HISCORES" home page, we intended to first pull out all players' information with XP scores, level, and rank columns. As I mentioned in the explore data section, there are 29 skills, "overall" included. Each page for each skill had 25 players, and there are 20,000 - 60,000 pages for each skill. This indicates 25 * 20,000 players' information from one individual skill that

we need to pull. Nonetheless, the IP address restricted us that we are only allowed to pull 264 pages from one IP address. If we are willing to pull more information, we need either change our location or wait until the next day. On the other hand, there are 29 skills' data that need to be collected, which indicates that there are at least 20,000(pages) * 25(players/page) * 29(skills) data points that need to be collected, it also gave us a hard time to joining all skills across the data.

The second limitation that we run into is to manually generated the URLs. In order to get individual skills for each player, we need to click the player itself from the "HISCORES" home page and go to a different URL page called "COMPARE PLAYERS".

*Example Figure 1: HISCORES home page*


*Example Figure 2: COMPARE PLAYER page.*

The “COMPARE PLAYERS” page includes two players' data with their individual skills. In the *NewScrapper.R*, where we web scrapped all players’ data with their individual skills. However, the web scraper can only pull around 1k URLs at a time, which indicates that we can only get around 2k players’ data from one run of the NewScrapper script. In this case scenario, we need to manually set up the number in the for loop from the NewScrapper.R file and run the file multiple times in order to get enough players’ data points.

## 2. Use more complicated models

Secondly is our machine learning model. Besides linear regression analysis and Xgboost, we also intended to use Bayesian Network and Neural Network. Bayesian Network is a probabilistic graphic model for representing knowledge about an uncertain domain. Each node corresponds to a random variable, and each edge represents the conditional probability for the corresponding random variables (Science Direct, 2019). However, due to the highly correlated issue of our dataset, Bayesian Network not only won't work perfectly on our dataset but also increases the bias of the results. Like the Bayesian Network, Neural Network tends to find the hidden pattern and correlations in the raw dataset. The primary issue of our raw dataset is that each variable is highly correlated. For example, when a player's skill level gets higher, there is no doubt that the XP score will increase, same for the rank. Highly correlated datasets affect the accuracy of Bayesian Network and Neural Network results.

## References

*Bayesian network*. Bayesian Network - an overview | ScienceDirect Topics. (n.d.). Retrieved May 3, 2022, from https://www.sciencedirect.com/topics/mathematics/bayesian-network#:~:text=A%20Bayesian%20network%20(BN)%20is,corresponding%20random%20variables%20%5B9%5D.

Fulton, M. (2020, December 2). *RuneScape: What it is and how to play*. Lifewire. Retrieved May 3, 2022, from https://www.lifewire.com/runescape-what-it-is-and-how-to-play-4129385

Mugdi, mugdimugdi 22744 silver badges1212 bronze badges, & user18815063user18815063 2133 bronze badges. (1969, December 1). *Create shap plots for Tidymodel objects*. Stack Overflow. Retrieved May 3, 2022, from https://stackoverflow.com/questions/71662140/create-shap-plots-for-tidymodel-objects

CyberguilleCyberguille 51922 gold badges99 silver badges2020 bronze badges, fbtfbt

47444 silver badges77 bronze badges, & abalterabalter 85077 silver badges1818 bronze badges. (1962, April 1). *How to get the value of mean squared error in a linear regression in R*. Cross Validated. Retrieved May 3, 2022, from https://stats.stackexchange.com/questions/107643/how-to-get-the-value-of-mean-squared-error-in-a-linear-regression-in-r

DataTechNotes. (2019, June 26). *Regression example with xgbregressor in python*. Regression Example with XGBRegressor in Python. Retrieved May 3, 2022, from https://www.datatechnotes.com/2019/06/regression-example-with-xgbregressor-in.html

Wang, X. (2021, August 19). *How to interpret and explain your machine learning models using Shap Values*. Medium. Retrieved May 3, 2022, from https://m.mage.ai/how-to-interpret-and-explain-your-machine-learning-models-using-shap-values-471c2635b78e